LA-UR -81-1576

CONF-810523--5

TITLE: DESIGN OF THE RSX11M Q SYSTEM

MASTER

AUTHOR(S): James F. Harrison, MP-1
Thomas Kozlowski, MP-1
Richard A. Floyd, MP-1
James F. Amann, MP-10
Gail T. Anderson, MP-1
Michael A. Oothoudt, MP-1
Dennis G. Perry, MP-1

SUBMITTED TO: Topical Conference on Computerized Dat. Acquisition in Particle and Nuclear Physics, Oak Ridge, TN, May 28-30, 1981

University of California

LOS ALAMOS SCIENTIFIC LABORATORY
Post Office Box 1663 Los Alamos, New Mexico 87545
An Affirmative Action/Equal Opportunity Employer

Form No. 836 R3
St. No. 2629
12/78

UNITED STATES
DEPARTMENT OF ENERGY
CONTRACT W-7405-ENG. 36

DESIGN OF THE RSX11M Q SYSTEM

James F. Harrison, Thomas Kozlowski, Richard A. Floyd, James F. Amann,
Gail T. Anderson, Michael A. Oothoudt, and Dennis G. Perry*
Los Alamos National Laboratory
Los Alamos, NM 87545

Abstract

The present LAMPF Q data acquisition/analysis software system is being converted to run under the RSX-11M operating system. Major subsystems have been redesigned to be compatible with RSX-11M and to provide enhanced functionality. In addition, two new major subsystems have been designed: a test package subsystem and a real-time parameter array subsystem.

Summary

The Q data acquisition/analysis system for use under the RSX-11D operating system on Digital Equipment Corporation (DEC) PDP-11 minicomputers has been in use at the Los Alamos Meson Physics Facility (LAMPF) for several years.[1-4] During this time the need for additional capabilities has been pointed out by users. This has prompted an effort to develop an enhanced version of Q. The new version of Q runs under the RSX-11M operating system. This paper describes the current effort and the additional features of RSX-11M Q.

The subsystems of Q (shown in Figure 1) are: 1) data collection, 2) data replay, 3) histogramming, 4) histogram support and display, 5) dotplotting, 6) test package, and 7) real-time parameter array. The test package and real-time parameter array are new subsystems to Q.

In the data collection subsystem, new features include: 1) an event size limited only by the size of the buffer in the CAMAC branch driver (the BiRa MBD) - approximately 3000 16-bit words, 2) filtering of raw data before taping, 3) taping of software generated data, and 4) better user access to the system.

The data replay subsystem shares most of the new features in the data collection subsystem, including generation of output tapes containing "filtered" input data.

The redesign of the histogramming subsystem is based on an RSX-11M I/O driver. It has a histogram block mode so that all histograms in a block can be updated with a single call to the driver in order to minimize the histogramming time overhead.

A user-written test package has been in use by some users at LAMPF. A redesigned package is part of the standard RSX-11M Q system. This subsystem provides single parameter cuts, two parameter cuts, bit tests, etc., on the data. Tests can be grouped together in various ways. The user and the histogramming subsystem can then use the results of these tests to determine an action to be taken. Histograms can be "gated" by the results of the tests.

The histogram support and display subsystem provides an interface with the test package so the user can display cuts and modify them using cursor input. Also, flexible histogram setup facilities are now available from the keyboard.

The real-time parameter array subsystem provides keyboard tasks to initialize and modify real and integer array elements within the analyzer task. These array elements can then be used in the analyzer for whatever the experimenter deems appropriate.

This version of Q provides many new features which are needed to support more complex experiments which are being planned at LAMPF and which make it easier for the user to develop more complex data acquisition and analysis software.

Introduction

History

The Q system was created to provide a general and relatively complete data acquisition system for use with experiments at LAMPF. It was written to run on

Figure 1. RSX-11M Q data acquisition/analysis system showing the major data flow between subsystems. The three top center boxes represent the data collection/data replay subsystem.

*Work supported by the U.S. Department of Energy.

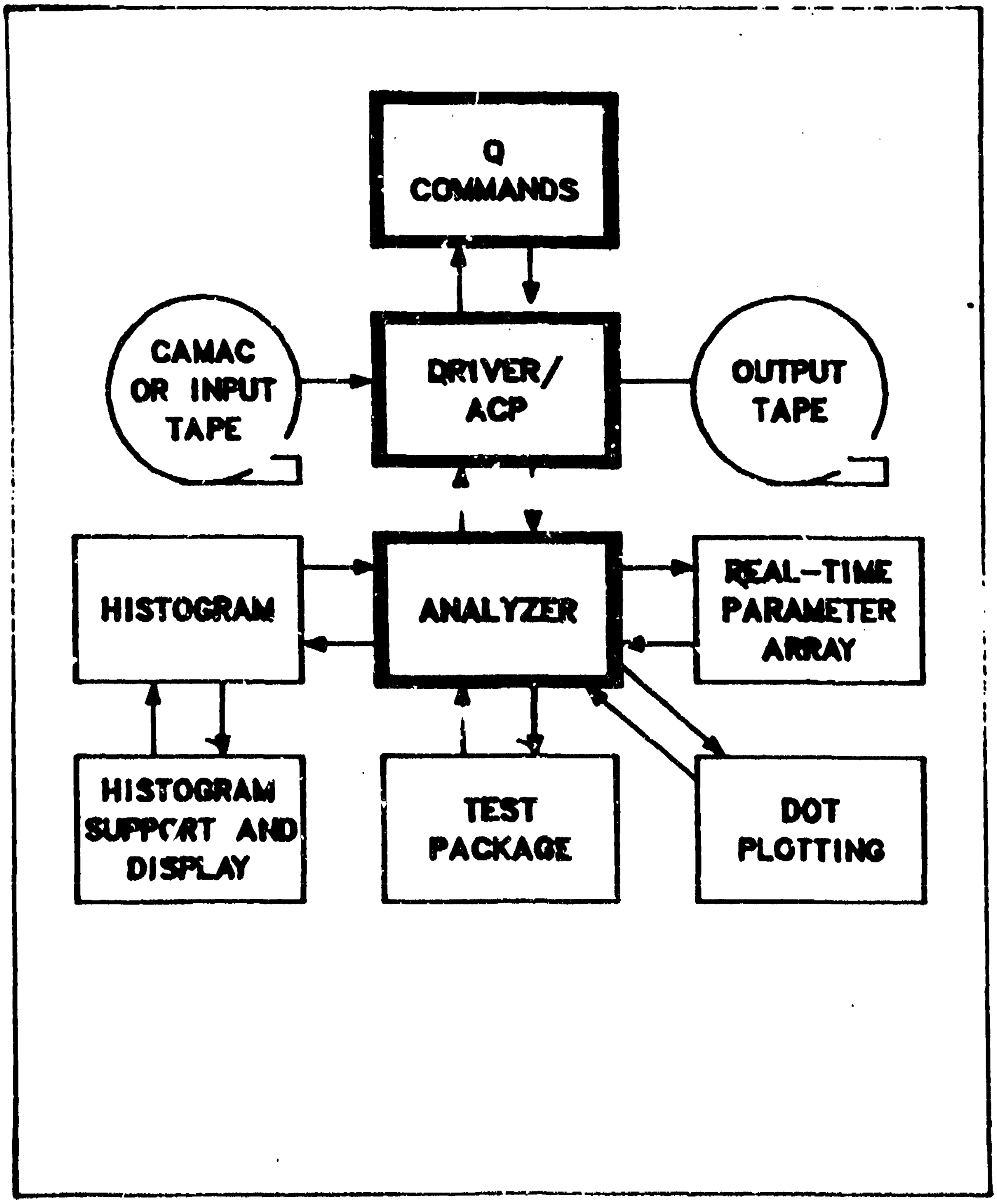
Q COMMANDS
CAMAC OR INPUT TAPE
DRIVER/ ACP
OUTPUT TAPE
HISTOGRAM
ANALYZER
REAL-TIME PARAMETER ARRAY
HISTOGRAM SUPPORT AND DISPLAY
TEST PACKAGE
DOT PLOTTING

FIGURE 1

DEC PDP-11 computers[5] under the RSX-11D operating system and has been in use since about 1975. At that time, RSX-11D was the most suitable operating system for accomplishing the desired objectives. Recently (1979) a multiuser version of Q replay was written to run under the VMS operating system on the DEC VAX-11/780 computer.

When the RSX-11M operating system first became available, some studies and preliminary work were done to produce an RSX-11M version of Q. This effort was not fruitful because of the unforseen size of the project (RSX-11M was not as similar to RSX-11D as originally claimed by DEC) and because of the manpower requirements.

Because of the retirement of the RSX-11D operating system by DEC, the incentive to convert the Q system to RSX-11M increased. Furthermore, in the intervening years RSX-11M has improved to the point that it now has most of the features of RSX-11D and some features not present in RSX-11D. Additional incentive arises from several deficiencies that have become apparent in approximately six years of using the Q system. For these reasons, the conversion to RSX-11M is being carried out.

Overview of the Basic Q - System

The Q system is based on the standard data acquisition computer system at LAMPF. This system includes a PDP-11 CPU with a memory management and hardware floating point units, 256K bytes of memory, at least 4 Mb bytes of disk storage, magnetic tape, a terminal, a graphics CRT, a printer/plotter, and an MBD branch driver. The MBD is a programmable branch driver. It contains sufficient memory (up to 8K bytes) for extensive data acquisition code, and a large data buffer.

A LAMPF-designed CAMAC "trigger module" provides a standard interface to the experiment. The module allows 32 trigger inputs which are priority-ordered by the module. An event trigger received at an enabled input generates a LAM which causes the MBD data acquisition code to execute. The MBD reads the highest priority trigger number, which has been encoded by the module, and carries out the appropriate user-specified CAMAC and MBD operations for that trigger number. The data from several events is stored in the MBD's internal buffer. On command, or when it is filled, the buffer is dumped to buffer in the PDP-11. The data in the PDP-11 buffers can then be written to tape and distributed for processing. The user provides a subroutine (usually written in FORTRAN) for each defined event. The PDP-11 analysis program ("analyzer task") calls the appropriate processor as it scans through the events in the buffers. For replay, the PDP-11 data buffers are filled from the Q format input tape instead of from the MBD.

This basic system is part of a system of various control and auxiliary tasks that constitute the whole Q system.

Current Capability and Required Capability

In the six years of its use, the Q system has been adequate for its purpose of providing a general and easy to use means of acquiring CAMAC data, recording of the data on tape and providing a simple interface for on-line and off-line processing of experimental data.

Some limitations of the RSX-11D Q system are the event size (limited to 254 data words), difficulty of recording user (software) generated data on the output tape, some problems with tape buffers (size and format), and difficulties in handling some experimental control functions. Some additions to the RSX-11M Q system which increase its usefulness are tools that provide more effective on-line interaction with the system and with user-written analyzers, and more flexible event distribution and processing. The RSX-11D Q deficiencies and limitations result in wasted time and effort in finding ways around them and in the writing of software by the user which might better be provided with the Q system.

Considerations that favor conversion from RSX-11D to RSX-11M are the following: 1) VAX/VMS supports an RSX-11M compatibility mode, 2) the Q system could be used outside of LAMPF where there are many RSX-11M installations, 3) there would be access to the large amount of RSX-11M software available in the RSX user community, 4) RSX-11M supports new hardware that RSX-11D does not, and 5) RSX-11M has some very useful features which are not available under RSX-11D, e.g., memory management directives and memory resident overlays.

Data Collection Subsystem

Components

Major components of the data collection subsystem are: 1) QAL - MBD code, 2) the data analysis task (analyzer), 3) an I/O driver - ACP, and 4) the keyboard command tasks. Figure 2 shows the major data and control flows for this subsystem.

The QAL[6] language is used to define the CAMAC operations to be performed when a particular event trigger is received. It remains essentially the same as under RSX-11D. QAL produces an object module containing event specification tables that is read into the MBD. Those event specifications are used by the table-driven interpretive code in the MBD to perform the specified CAMAC and other operations when an event trigger has been received by the LAMPF trigger module.[3]

The analyzer is a nonprivilged RSX-11M task that consists of standard Q system support routines and user written subroutines. The Q system support routines perform the system initialization, the retrieval of data buffers, the distribution of events to the processing subroutines, and synchronization with the rest of the Q system. The experiment-specific user written part consists largely of subroutines to process the data for individual events. These are generally written in FORTRAN. The analyzer task also contains the data buffers and the Q data base.

The underlying system is implemented as an RSX-11M driver (part of the executive) and an Ancillary Control Processor (ACP) which is a privileged task that augments the driver. The ACP records the data buffers on magnetic tape, contains the MBD interrupt service routine, and synchronizes and controls the operation of the Q system. The I/O driver is the same, but different ACPs are used for data acquisition and replay.

The Q system is controlled from a terminal by keyboard command tasks. These tasks are similar to those for RSX-11D Q with the addition of the QCM

Figure 2. Data and control flow diagram for the data collection/data replay subsystems. Single lines represent I/O requests and double lines represent data flow.

command which allows entering comment lines into a special comment record on the output tape. Some of the other commands have different user interfaces and somewhat different functionalities. The QBE and QKI commands are used to start up the Q system including the loading of the MBD code and to terminate the system. QBE is also used to pass the CAMAC address of the trigger module and the MBD buffer size to the system. The run control commands QRU, QFI, QSU, QRE, are used to start, stop, suspend, and resume runs respectively. The taping control commands QET, QIT, QNT, QCT, enable taping, inhibit taping, mount new tapes, and close out output tapes. A command parameter is used to designate whether an input or output tape is being referred to. The event control commands QEN, QPA, and QDE are used to enable/disable event triggers, declare events as "must", "may" or "no" process, and trigger events at the trigger module. "Must" process events will always be processed, "may" process events will be processed if there is sufficient buffer space for new data, and "no" process events will never be processed. The QST command reports the status of the Q system. Various status options are available which make much more status information available than in the RSX-11D Q system.

New Features

The user has more control in the RSX-11M Q system over the data which is written to tape. The option is available of writing to tape only those events which are found to be acceptable by the processing routine. This procedure is referred to as "filtering". This is practical because the raw data buffers (input) and user data buffers (output) are in the analyzer task providing fast and easy access.

The event size is limited only by the size of the MBD buffer (a maximum of approximately 3000 16-bit words) or the size of the PDP-11 buffer, whichever is smaller. The sizes of the MBD and PDP-11 buffers are defined by the user. The user can also specify the number of PDP-11 buffers of each type (raw data and user data buffers).

Since the buffers are present in the analyzer task, the user can be allowed more freedom in accessing event data. In particular, it is possible to retrieve portions of an event for a "quick look" before deciding to retrieve the remainder. Standard subroutines are provided to move an event or part of an event into a local array, move a local array into a user data buffer for output, and move an event from a raw data buffer into a user data buffer. External tasks are able to write data to the output tape by issuing an I/O request to the Q device driver.

A function is provided that allows the analyzer to retrieve any element from the Q data base. For instance, the user may want to determine if there is time available to carry out more extensive calculations on particular event data. Both read and write access to the data base is provided for external tasks through an I/O request to the Q device.

The RSX-11D Q system suffers from a lack of provisions for user control and intervention at certain points such as starting, stopping, suspending, resuming runs, and on encountering the end of tape. RSX-11M Q provides more optional tasks and subroutines that may be invoked at critical points. Tasks with predefined standard names may be requested to run before and after accessing the hardware (LAMPF trigger module) when stopping, starting, suspending, and resuming runs. In addition, subroutines with standard names may be called in the analyzer task whenever a control event (stop, start, suspend, or resume) is encountered in the input data stream. (Each time a control function such as stopping or starting a run is done a special control event is put into the input data stream.) Predefined tasks are called when an end of tape mark is sensed on an output tape and when an end of volume mark is encountered on an input tape (replay).

More flexibility has been added to the MBD code. As mentioned above, the user can tailor the MBD buffer size to his particular needs. A "FLUSH" QAL statement has been added that causes the MBD buffer to be dumped to the PDP-11 when the acquisition of the current event is complete. The PDP-11 buffer that the "flushed" event ends up in then defined as complete and ready for processing. This feature is needed to provide quick analyzer task response to events that implement experiment control functions. The RSX-11D QAL "DUMP" statement, which simply caused the MBD buffer to be dumped to the PDP-11, is also available. Since PDP-11/70s and PDP-11/44s, which are 22-bit bus machines, will be used at LAMPF in the near future for data acquisition, the MBD (an 18-bit Unibus device) support software has been modified to use the Unibus Mapping Registers of these machines.

Data Replay Subsystem

The main difference between the data replay subsystem and the data collection subsystem is that the input stream to the Q system is from data tapes instead of CAMAC. In this case, the interrupt service

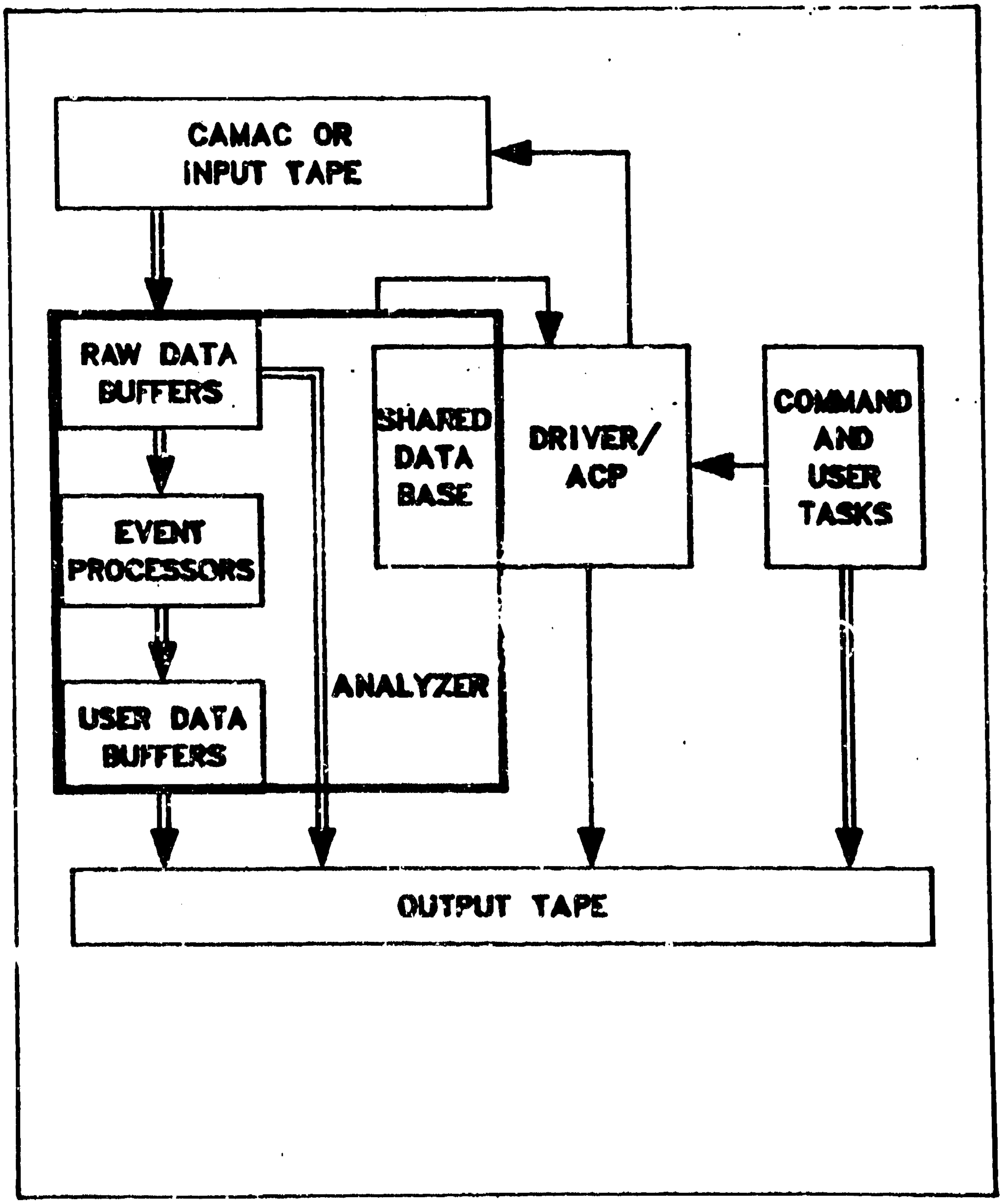
CAMAC OR INPUT TAPE
RAW DATA BUFFERS
EVENT PROCESSORS
USER DATA BUFFERS
ANALYZER
SHARED DATA BASE
DRIVER/ ACP
COMMAND AND USER TASKS
OUTPUT TAPE

FIGURE 2

routine to handle MBD interrupts is replaced with a routine to read the input data tapes. Also, there is no QAL counterpart in the replay subsystem.

The Q system was designed so that at a later time multiuser replay and also concurrent replay and data acquisition can be easily implemented. Each user would use a separate unit of the Q device and would have his own analyzer task and his own copy of the appropriate ACP task.

The replay subsystem keyboard commands have the same functionality as the data collection commands. To distinguish them they begin with the letter "X" instead of "Q".

Histogramming Subsystem

The Q histogram management subsystem allows the user to create, delete, store data in, and retrieve data from histograms. In converting the RSX-11D subsystem to RSX-11M, several new features have been added:

- A block mode of histogram entry is provided which allows several histograms to have data entered by a single subroutine call, simplifying the logic of the user task and decreasing overhead.
- Gating of histogram entry by logical tests is allowed, either through the Q test package, or by a user-supplied test subsystem.
- True multiuser histogramming with protection of data is supported.
- The subsystem no longer limits the number of histograms that may be defined, nor does it limit how much memory may be used for histogram storage.
- Each histogram bin may contain from 0 to 65535 counts. Up to 2048 bins in each histogram may "overflow" into special counters which allow the maximum count to be 4,294,967,295.
- Data to be histogrammed may be an integer in the range -32768 to +32767. (The data value is converted to a bin number.)
- Arbitrary positive integers may be added to bins, either event-by-event or as an array put into a series of bins.

The new block histogram entry mode is probably the most important improvement to the subsystem. Under the RSX-11D subsystem the user scattered calls to the histogram entry routines almost at random throughout his code. In order to maintain histogram entry speed, a hidden buffering scheme was required that caused confusion when the buffers were not properly flushed. In the RSX-11M subsystem, the user puts block histogram entry calls in his code at locations where there is a significant amount of data that needs to be histogrammed, e.g., where raw data is received, where kinematics calculations are completed, where valid coincidences of proper particle types are determined, etc. Furthermore, the user code does not need to know which histograms are defined, since the subsystem does this for the user.

The central control module in the subsystem is the RSX-11M I/O driver HM. An I/O driver was chosen to make use of the standard interface supplied by DEC between user and privileged code. (Privileged code is required for fast access to data stored outside of the user task.) Furthermore, in the future we hope to implement histogram storage in special peripheral memory (a "histogram box") for which an I/O driver will be required. Communication with HM is via standard operating system I/O requests with separate I/O requests provided to implement each of the actions required of HM. Since use of I/O requests is complicated for the typical user, a set of subroutines and tasks is provided to interface to HM. The subroutines are:

- HSTBLK performs block histogram entry.
- HSTCLH, HSTCLB, and HSTCLA, respectively, clear (zero) one histogram, a block of histograms or all histograms owned by a user.
- HSTDEH, HSTDEB, and HSTDEA, respectively, delete one histogram, a block of histograms, or all histograms owned by a user.
- HSTHPB retrieves the parameters defining a histogram.
- HSTINI and HSTTIN initialize communication with HM for a task doing histogram entry.
- HSTMEH allows multiple entry histograms, i.e., enters data into several bins of a single histogram with a single subroutine call.
- HSTOWN retrieves a list of all owners currently known to HM.
- HSTRET retrieves the data from a histogram. Data for a histogram may be retrieved by any task in the system, making data display simple.
- HSTSU sets up (creates) a new histogram.
- HSTUPD allows the user to put an array of data into a histogram. The array may either overwrite or be added to the current histogram contents.

The interface tasks allow the user to control the subsystem from the keyboard:

- HBE makes a user known to HM and allocates the required data bases.
- HOF removes a user from HM and releases the data bases.
- HPK allows the user to inspect the data storage region. A future version will also allow the user to manipulate the storage areas to reduce fragmentation of memory.

HM maintains one data base to describe each histogram owner and three other data bases for each owner:

- The owner table contains a list of users who have been made known to HM. Owners are distinguished by their unique User Identification Code (UIC). The owner table also contains a pointer to each owner's block table.
- The block table contains a list of pointers to the first histogram descriptor defined for each histogram block. The histogram descriptors in

the block form a linked list with each histogram pointing to the next one in the list.

- A histogram descriptor exists for each defined histogram and contains all the definition information as well as special statistics about the histogram (e.g., how many histogram entries failed because the data was outside the bounds of the histogram). The descriptors are stored contiguously with the data.

- The data storage region cannot be in the user task, which is limited to 64K bytes. The data is kept in an RSX-11M memory management region that is maintained by HM (direct access by the user task would be very slow). A separate region is maintained for each owner.

The subsystem implements almost all functions requested by the users and requires considerably less space in memory and in the user task than the old RSX-11D subsystem. The only speed-critical operation in the subsystem is histogram entry which must be done on an event-by-event basis. The time for histogram entry in milliseconds is approximately

$$T = 2.72 + 0.14*n$$

on a PDP-11/34 where n is the number of histograms in the block.

Histogram Support and Display Subsystem

The histogram support and display subsystem[7] allows the user keyboard access to most of the functions provided by the histogram support subsystem. Experimenters rarely have the time or facilities to develop generalized histogram setup and display programs. Therefore, this subsystem is provided to allow effective use to be made of the histogramming subsystem. Tasks are provided which allow the user to create, clear, and delete histograms (HSU); list information about histograms currently defined (HLI); print histograms (HPR); and display histograms in a number of different formats on a TEKTRONIX 4000 series (or compatible) terminal (HPL). Histograms may also be saved into a file (HSV and HTR) for later display and analysis.

HSU (Histogram Set Up) is the only task in the histogram support and display subsystem which did not exist under RSX-11D. Histograms may be created using HSU in a number of different ways:

- The user enters histogram parameters from the keyboard. When a histogram is completely defined, it is created.

- The user specifies the name of a histogram which already exists, along with parameters to be changed (e.g., change the X range of histogram TDC12 to 0 through 1024). HSU deletes the existing histogram and recreates it with the new parameters.

- The user creates a file describing histograms to be set up and then specifies it as input to HSU (useful when a number of histograms are always defined together).

- The user specifies the name of a file containing saved histograms (which HSU recreates in core) or specifies that histograms be set up from the descriptions given in the disk data base maintained by HSU (described below). This allows histograms defined in earlier runs to be set up quickly (useful when replaying a run or recovering from a system crash).

To minimize the amount of typing required to create histograms, HSU allows defaults to be defined for all histogram parameters. In addition to creating histograms, HSU may be used to clear and delete histograms. The clear and delete functions allow the user to clear/delete a single histogram, all histograms in a given block, or all histograms belonging to the user. HSU creates and maintains a disk data base used by other tasks in the subsystem (e.g., HPL uses the data base to store plotting information for each histogram). Entries are added to/deleted from the data base as histograms are created/deleted.

HPL (Histogram PLotting) is similar to the RSX-11D version of HPL but with the addition of a facility which allows the user to set up the data indices and limits for tests in the test package subsystem. To set up a "window test" on a single parameter, the user displays a histogram of the parameter, specifies the number of the test to be set up, and then uses the hardware cursor to mark the lower and upper limits of the window (or, instead of using the cursor, the user may input the values of the limits directly). To set up a "box test" on two data parameters, the user displays the appropriate two parameter histogram, specifies the number of the test being defined, and then uses the cursors to box the area of interest. HPL also displays the test number and test limits when displaying histograms of parameters which have had tests set up on them in this fashion.

HSV (HIstogram SaVing) has been modified to allow the user to save test descriptors and test results from the test package subsystem and to save the contents of the real-time parameter array in a histogram save file. Modifications to the histogram save file format allow these to be distinguished from histograms and to be retrieved at a later time.

HLI (HIstogram LIsting -- HIS under RSX-11D), HPR (HIstogram PRinting), and HTR (Histogram TRansfer) have not changed significantly in the conversion from RSX-11D to RSX-11M.

Dotplotting Subsystem

The dotplotting subsystem[7] allows the user to display live two parameter scatter plots using a TEKTRONIX 4000 series terminal screen as 'storage'. These plots are typically used during experimental setup and for quick looks at data during the course of a run. A single task (HDO) is used to define and control plots and maintain a disk data base describing plots that have been defined (the presence of this data base allows plots to be redisplayed at a later time by simply specifying the plot name). Subroutines are available to allow the user task to initialize the dotplotting subsystem and to plot points.

In converting to RSX-11M, two major changes have been made to the dotplotting subsystem:

- When defining a dotplot, the user is able to specify that a test package test be passed/not passed before plotting a point on the screen.

- The user is able to set up (using HDO in a manner similar to the one described above for HPL) test package tests on data pairs. Tests set up in this way are displayed when plots of the data pair are displayed.

Test Package Subsystem

The test package provides a tool to examine data on an event-by-event basis in a fast and flexible manner. The results of these tests are available to the user for decision making in the course of event analysis and may be used by the histogramming and dotplotting subsystems as well. In addition, counters are kept for each test to provide the user with global information about the data analysis.

In keeping with the goal of maximum flexibility only a bare minimum of detail need be specified in the analyzer at compile time. This consists primarily of a call to an initialization subroutine and a call to the general test execution subroutine at the appropriate place in the analysis for each block of tests to be performed. All the details about the tests are contained in a descriptor file which is specified by the user at run time.

The tests are of two basic types, data tests which examine the data, and logic tests which examine logical combinations of previous test results. These generally correspond to FORTRAN ARITHMETIC IF and LOGICAL IF statements, respectively. Both types of tests may be freely intermixed within each block of tests. Each test is given a numeric label so that it may be referenced by the logic tests. The order in which the tests are executed is determined by the ordering in the descriptor file. This allows the user to add tests without a tedious renumbering procedure. Some of the test functions that are implemented in the test package subsystem are the following:

- Window Test - This test is true if a specific data word is both greater than some lower limit and less than or equal to some upper limit.

- Bit Test - This test is true if a specified bit in a given data word is set to 1.

- Indirect Window Test - This test has the same requirements as in the standard Window Test, but data word and limits are provided from cursor input on a histogram.

- Box Test - This test is true if two specified data words are both greater than their individual lower limits and less than or equal to their individual upper limits. Both the data word indices and the limits are provided via cursor input from a dotplot or two parameter histogram plot.

- And Test - This test is true if and only if all the tests specified to be true are actually true and all tests specified to be false are actually false.

- Inclusive Or Test - This test is true if at least one of the tests specified to be true is actually true or (inclusive) at least one of the tests specified to be false is actually false.

- Majority Test - This test is true if at least N of all tests specified are in the desired state (either true or false).

The test package data base contained within the analyzer task is structured for rapid internal access. It contains the latest 8-bit logical result and a 32-bit counter for each test, a 32-bit counter for each block, and pointers to the beginning of each of these arrays. It also contains a set of pointers to the beginning of each block of tests and to the data specifying each test, both organized so as to minimize execution time. This data base is accessed from only two external tasks. The test setup task (TSU) reads the test descriptor file and updates the internal data base appropriately. The task TSTDDB reads the internal data base and creates a disk file which contains a snapshot in time of it. All other external tasks needing information about the tests access the disk file.

The test package provides the users with a set of functions, subroutines, and external tasks which allow the user extensive access to the test data base while maintaining the integrity of the system. From within the analyzer task, in addition to the initialization and execution subroutines previously mentioned, the user can determine the latest result for any test, retrieve and/or zero the test and block counters, and zero out the test results for any or all blocks. In addition, a facility is provided for the user to incorporate information from his own tests into the test package via the concept of a "user test", and a corresponding subroutine.

As mentioned above, all access to the test data base from outside the analyzer task occurs through the tasks TSU and TSTDDB. Subroutines are provided to 1) retrieve and/or zero the test and block counters using TSU and TSTDDB and 2) retrieve or modify test limits (again using TSU and TSTDDB). Tasks are provided to list the counters for all tests on a specified output device and to decode the test data base as a diagnostic aid should that data base become corrupted.

Real-Time Parameter Array Subsystem

The real-time parameter array subsystem allows the Q user to change the values of variables in his analyzer task while it is running. It also provides a disk backup of the parameters.

The array of parameters which may require real-time access by the user is held in a memory management region. The user interface to the array is through interactive keyboard commands. The interface to the parameter array by the analyzer task and other user tasks is through calls to subroutines which create and map to the memory management region.

The parameter array consists of four array sections to hold 16-bit integer, 32-bit real, 32-bit integer, and 64-bit real values. In order to conserve analyzer task virtual address space, the user may optionally choose to include in the same region (following the parameter array) the test package subsystem data base and/or the Q data base. The whole region may not be larger than 16K words.

The parameter array is initially set up when the analyzer calls subroutine PRMCRE which creates, attaches, and maps to a memory management region. The analyzer task then accesses the parameters through a FORTRAN common block.

The user may perform the following functions at a keyboard terminal: 1) initialize the values in the array from a backup file, 2) set values individually,

.e., R15=0.25, 3) define synonyms to use when eferring to individual parameters or groups of equential parameters, i.e., I21:=S7PED, 4) back up he values in the region to a disk file at any time, nd 5) examine or print out the whole parameter array r any portion of it. Also, the user may provide iles containing 68-character titles or descriptions f parameters to be used in listing parameters and alues.

Utilities

The utilities are not strictly part of the SX-11M Q system. They are a collection of tasks to acilitate development of the Q system, to assist the ser in producing the experiment-specific part of the ser's Q data acquisition system, and to provide iagnostic capabilities. These tasks include reprocessors to produce structured FORTRAN and ssembly language programs, a program to extract nformation from source files to produce ocumentation, programs to dump the contents of agnetic tape and of the MBD branch driver, and a rogram to initiate single CAMAC operations. The last rogram is particularly useful when debugging CAMAC ardware.

A new utility task is provided to print Q data apes in a readable format. Options are provided in his task to print beginning of run records, print by un number, print comments, produce a summary of the umber of each type of event in a run, and print arious levels of event data information.

Q Subroutine Library

The Q subroutine library is a collection of ubroutines which are used by the various Q ubsystems, and also subroutines which interface to he various Q subsystems. Providing a subroutine ibrary requires only one Q system file that links ith the user software, thus easing the user's task of roducing a Q data acquisition/analysis system for the ser's experiment.

Conclusion

The additions and improvements to the Q software ill allow it to support the wide range of experiments eing done at LAMPF now and in the future. This apability will be extended by future projects to mprove the QAL language and to support additional ardware. A Q data acquisition/analysis system unning under RSX-11M should be a viable and aintainable system for many years in the future.

References

1. Dennis G. Perry, "Distributed Systems Handle Data for Meson Research," Industrial Research and Development (December 1980).

2. Dennis G. Perry and Dale C. Sparks, "Data nalysis at LAMPF: A Hierarchical Approach," IEEE Trans. Nucl. Sci. NS-26, 4572 (1979).

3. Dennis G. Perry, "The Q System: An Experiment in Software Standardization for Data Acquisition," IEEE Trans. Nucl. Sci. NS-26, 4494 (1979).

4. Michael M. Minor, Sally Shlaer, and Richard F. Thomas, Jr., "A Software System for Data Acquisition in Nuclear Physics Experiments Using CAMAC," IEEE Trans. Nucl. Sci. NS-23, 459 (1976).

5. Dennis G. Perry, "The LAMPF Standard Data-Acquisition System," IEEE Trans. Nucl. Sci. NS-26, 4422 (1979).

6. Dennis G. Perry, "QAL: A Language for Nuclear Physics Data Acquisition," IEEE Trans. Nucl. Sci. NS-26, 4508 (1979).

7. Michael A. Oothoudt, "Data Display with the Q System, IEEE Trans. Nucl. Sci. NS-26, 4473 (1979).